



UNIVERSIDAD NACIONAL
AUTÓNOMA DE MÉXICO



FACULTAD DE ESTUDIOS
SUPERIORES ARAGÓN

Ingeniería en Computación

COMPILADORES



Tarea 8 Máquina virtual para ensamblador de risc v

Profesor: Marcelo Pérez Medel

Leonardo Olvera Martínez

Grupo: 2608

Fecha: martes 07 de mayo de 2024

Parte A.

Revise el código del programa “maquina virtual rv asm.py” y explique cada parte

```
File Edit Selection View Go Run Terminal Help
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

maquina virtual RV asm.py X mv rv tk.py
C:\Users\usuario\Desktop> Uni > 2024-2 > Compiladores > Tarea 8 > maquina virtual RV asm.py > ...
1 # Clase para poder crear mejor las etiquetas
2 class Etiqueta:
3     def __init__(self, nombre, linea):
4         self.nombre = nombre
5         self.linea = linea
6
7 # Función para detectar si es un caracter especial
8 def es_simbolo_esp(caracter):
9     return caracter in "+-*,.:!@#$%^&*(){}[]<>=<=>="
10
11 # Funcion para separar con los separadores (espacio, tabulador, salto de linea)
12 def es_separador(caracter):
13     return caracter in " \n\t"
14
15 # Funcion para reemplazar las comas por un espacio seguido de una coma
16 def tokenizar(cadena):
17     # Reemplaza las comas por un espacio seguido de una coma
18     cadena = cadena.replace(',', ' ')
19     # Divide la cadena por espacios y devuelve la lista de tokens
20     tokens = cadena.split()
21     return tokens
22
23 # Creamos un array
24 instrucciones = []
25
26 # Abrimos el archivo "prog1.txt" para leerlo
27 arch = open("prog1.txt", 'r')
28 # Leemos cada linea
29 for ren in arch:
30     # Si la linea leida es mayor a 2
31     if len(ren)>2:
32         # Tokenizamos la linea
33         datos = tokenizar(ren)
34         # Metemos los tokens al array que creamos
35         instrucciones.append(datos)
36 # Cerramos el archivo
```

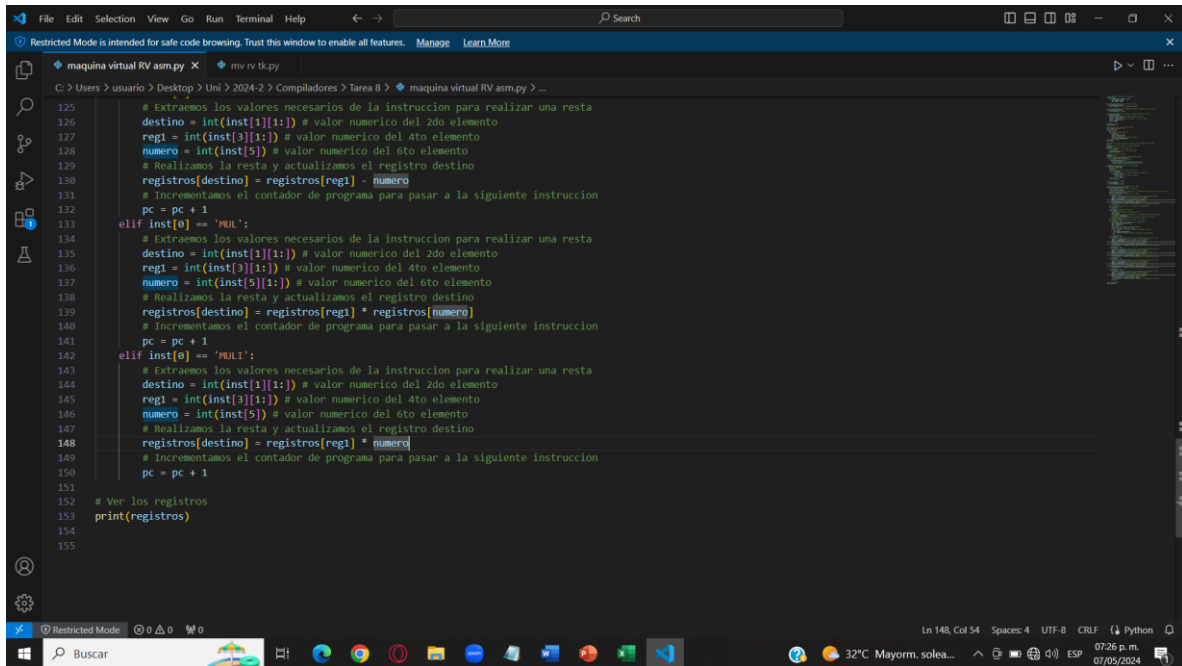
```
File Edit Selection View Go Run Terminal Help
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

maquina virtual RV asm.py X mv rv tk.py
C:\Users\usuario\Desktop> Uni > 2024-2 > Compiladores > Tarea 8 > maquina virtual RV asm.py > ...
35     instrucciones.append(datos)
36 # Cerramos el archivo
37 arch.close()
38
39 # Creamos una variable pc igual a 0 (para saber en que linea nos encontramos)
40 pc = 0
41 # Creamos un array que simulara ser nuestro procesador
42 registros = []
43 # usando una variable c hasta llegar a 32
44 for c in range(32):
45     # Agregaremos 0 32 veces a nuestro procesador
46     registros.append(0)
47
48 # encontrar todas las etiquetas
49 etiquetas = []
50 # Leemos todas las instrucciones tokenizadas que tenemos
51 for c in range(len(instrucciones)):
52     # Si encuentra un X (es una funcion)
53     if instrucciones[c][0] == "X":
54         # Creamos objetos de tipo etiqueta con lo siguiente
55         etiquetas.append(Etiqueta(instrucciones[c][1], c))
56
57 # Funcion para obtener la direccion de una etiqueta
58 def get_dir_etiqueta(etq):
59     # Creamos una variable
60     direccion = -1
61     # recorremos nuestro array con las etiquetas
62     for e in etiquetas:
63         # Si el nombre de uno de nuestros recorridos es igual a etq
64         if e.nombre == etq:
65             # Regresamos la linea donde se encuentra
66             return e.linea
67     # De no ser asi regresa -1
68     return direccion
69
70 # Imprimir todas las etiquetas
```

```
File Edit Selection View Go Run Terminal Help
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
maquina virtual RV asm.py x mv rv tk.py
C:\Users\usuario\Desktop> Uni\2024-2> Compiladores> Tarea 8> maquina virtual RV asm.py > ...
70 # imprimir todas las etiquetas
71 for e in etiquetas:
72     print(e.nombre, e.linea)
73
74 # Mientras en la lista de instrucciones no sea 'END'
75 while instrucciones[pc][0] != 'END':
76     # Almacenamos la fila actual en la variable inst
77     inst = instrucciones[pc]
78     # Imprimos el contador de programa (pc), el contenido del registro 2 y la instruccion actual
79     print(pc, registros[2], inst)
80     # Si la primera parte de la instruccion es 'ADDI'
81     if inst[0] == 'ADDI':
82         # Extraemos los valores necesarios de la instruccion para realizar una suma
83         destino = int(inst[1][1:]) # valor numerico del 2do elemento
84         reg1 = int(inst[3][1:]) # valor numerico del 4to elemento
85         numero = int(inst[5]) # valor numerico del 6to elemento
86         # Realizamos la suma y actualizamos el registro destino
87         registros[destino] = registros[reg1] + numero
88         # Incrementamos el contador de programa para pasar a la siguiente instruccion
89         pc = pc + 1
90     # Si la instruccion comienza con 'X' avanzamos e incrementamos PC
91     elif inst[0] == 'X':
92         pc = pc + 1
93     # Jump salta a la direccion indicada)
94     elif inst[0] == 'J':
95         # Obtenemos la etiqueta a la que vamos a saltar
96         etq = inst[1]
97         # Buscamos la direccion correspondiente a esa etiqueta en la tabla de etiquetas
98         pc = get_dir_etiqueta(etq)
99         # Imprimos un mensaje indicando que estamos saltando a esa direccion
100        print('saltando a', pc)
101    # Si la instruccion es una comparacion mayor o igual (BGE)
102    elif inst[0] == 'BGE':
103        # Extraemos los valores necesarios para la comparacion
104        op1 = registros[int(inst[1][1:])]
105        op2 = registros[int(inst[3][1:])]
106        num = int(inst[5])
107        etq = inst[1]
108        # Imprimos los valores que estamos comparando
109        print('comparando ', op1, 'con', num)
110        # Si la operacion es verdadera, saltamos a la etiqueta indicada
111        if op1 >= num:
112            pc = get_dir_etiqueta(etq)
113        # Si la operacion es falsa, simplemente avanzamos al siguiente paso
114        else:
115            pc = pc + 1
116    elif inst[0] == 'SUB':
117        # Extraemos los valores necesarios de la instruccion para realizar una resta
118        destino = int(inst[1][1:]) # valor numerico del 2do elemento
119        reg1 = int(inst[3][1:]) # valor numerico del 4to elemento
120        numero = int(inst[5][1:]) # valor numerico del 6to elemento
121        # Realizamos la resta y actualizamos el registro destino
122        registros[destino] = registros[reg1] - registros[numero]
123        # Incrementamos el contador de programa para pasar a la siguiente instruccion
124        pc = pc + 1
125    elif inst[0] == 'SUBI':
126        # Extraemos los valores necesarios de la instruccion para realizar una resta
127        destino = int(inst[1][1:]) # valor numerico del 2do elemento
128        reg1 = int(inst[3][1:]) # valor numerico del 4to elemento
129        numero = int(inst[5]) # valor numerico del 6to elemento
130        # Realizamos la resta y actualizamos el registro destino
131        registros[destino] = registros[reg1] - numero
132        # Incrementamos el contador de programa para pasar a la siguiente instruccion
133        pc = pc + 1
134    elif inst[0] == 'MUL':
135        # Extraemos los valores necesarios de la instruccion para realizar una resta
136        destino = int(inst[1][1:]) # valor numerico del 2do elemento
137        reg1 = int(inst[3][1:]) # valor numerico del 4to elemento
138        numero = int(inst[5]) # valor numerico del 6to elemento
139        # Realizamos la multiplicacion y actualizamos el registro destino
140        registros[destino] = registros[reg1] * registros[numero]
141        # Incrementamos el contador de programa para pasar a la siguiente instruccion
142        pc = pc + 1
143    else:
144        # Instruccion no reconocida
145        print('Instruccion no reconocida: ', inst[0])
146        pc = pc + 1
147
148 # Fin del programa
149 print('Fin del programa')
150
```

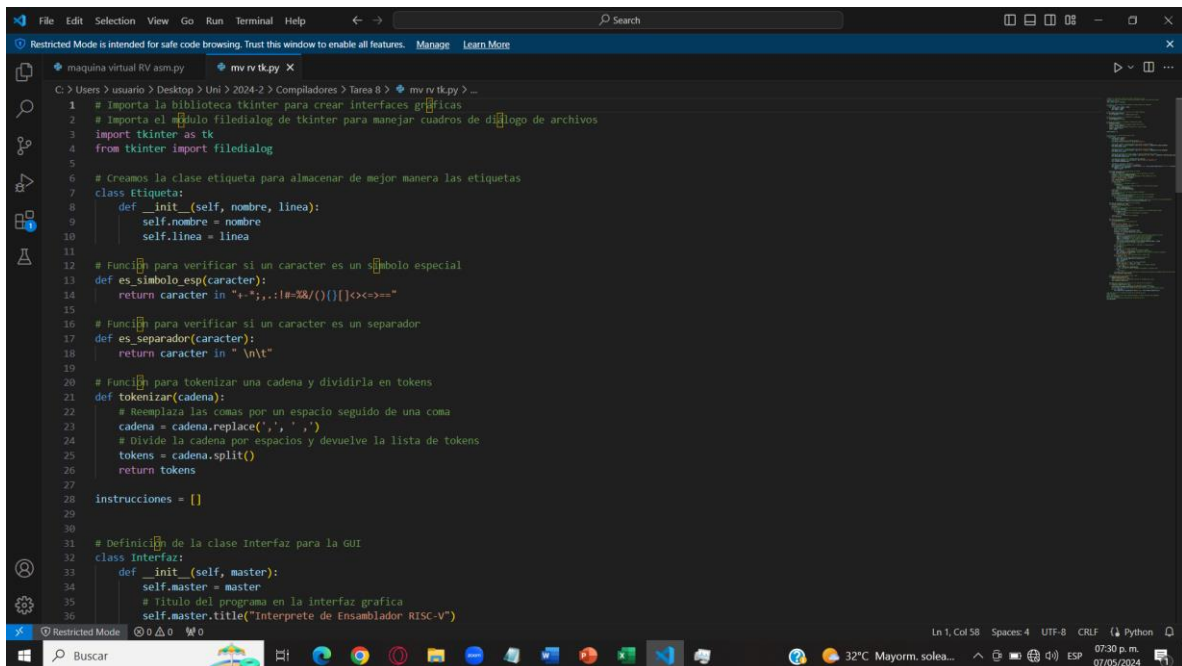
En esta parte del código fue agregado las instrucciones para “SUB” y “SUBI” además de “MUL” y “MULI”

```
File Edit Selection View Go Run Terminal Help
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
maquina virtual RV asm.py x mv rv tk.py
C:\Users\usuario\Desktop> Uni\2024-2> Compiladores> Tarea 8> maquina virtual RV asm.py > ...
101 # Si la instruccion es una comparacion mayor o igual (BGE)
102 elif inst[0] == 'BGE':
103     # Extraemos los valores necesarios para la comparacion
104     op1 = registros[int(inst[1][1:])]
105     num = int(inst[5])
106     etq = inst[1]
107     # Imprimos los valores que estamos comparando
108     print('comparando ', op1, 'con', num)
109     # Si la operacion es verdadera, saltamos a la etiqueta indicada
110     if op1 >= num:
111         pc = get_dir_etiqueta(etq)
112     # Si la operacion es falsa, simplemente avanzamos al siguiente paso
113     else:
114         pc = pc + 1
115     elif inst[0] == 'SUB':
116         # Extraemos los valores necesarios de la instruccion para realizar una resta
117         destino = int(inst[1][1:]) # valor numerico del 2do elemento
118         reg1 = int(inst[3][1:]) # valor numerico del 4to elemento
119         numero = int(inst[5][1:]) # valor numerico del 6to elemento
120         # Realizamos la resta y actualizamos el registro destino
121         registros[destino] = registros[reg1] - registros[numero]
122         # Incrementamos el contador de programa para pasar a la siguiente instruccion
123         pc = pc + 1
124     elif inst[0] == 'SUBI':
125         # Extraemos los valores necesarios de la instruccion para realizar una resta
126         destino = int(inst[1][1:]) # valor numerico del 2do elemento
127         reg1 = int(inst[3][1:]) # valor numerico del 4to elemento
128         numero = int(inst[5]) # valor numerico del 6to elemento
129         # Realizamos la resta y actualizamos el registro destino
130         registros[destino] = registros[reg1] - numero
131         # Incrementamos el contador de programa para pasar a la siguiente instruccion
132         pc = pc + 1
133     elif inst[0] == 'MUL':
134         # Extraemos los valores necesarios de la instruccion para realizar una resta
135         destino = int(inst[1][1:]) # valor numerico del 2do elemento
136         reg1 = int(inst[3][1:]) # valor numerico del 4to elemento
137         numero = int(inst[5]) # valor numerico del 6to elemento
138         # Realizamos la multiplicacion y actualizamos el registro destino
139         registros[destino] = registros[reg1] * registros[numero]
140         # Incrementamos el contador de programa para pasar a la siguiente instruccion
141         pc = pc + 1
142     else:
143         # Instruccion no reconocida
144         print('Instruccion no reconocida: ', inst[0])
145         pc = pc + 1
146
147 # Fin del programa
148 print('Fin del programa')
149
```



```
125 # Extraemos los valores necesarios de la instruccion para realizar una resta
126 destino = int(inst[1][1:]) # valor numerico del 2do elemento
127 regi = int(inst[3][1:]) # valor numerico del 4to elemento
128 numero = int(inst[5]) # valor numerico del 6to elemento
129 # Realizamos la resta y actualizamos el registro destino
130 registros[destino] = registros[regi] - numero
131 # Incrementamos el contador de programa para pasar a la siguiente instruccion
132 pc = pc + 1
133 elif inst[0] == 'MUL':
134     # Extraemos los valores necesarios de la instruccion para realizar una resta
135     destino = int(inst[1][1:]) # valor numerico del 2do elemento
136     regi = int(inst[3][1:]) # valor numerico del 4to elemento
137     numero = int(inst[5][1:]) # valor numerico del 6to elemento
138     # Realizamos la resta y actualizamos el registro destino
139     registros[destino] = registros[regi] * registros[numero]
140     # Incrementamos el contador de programa para pasar a la siguiente instruccion
141     pc = pc + 1
142 elif inst[0] == 'DIV':
143     # Extraemos los valores necesarios de la instruccion para realizar una resta
144     destino = int(inst[1][1:]) # valor numerico del 2do elemento
145     regi = int(inst[3][1:]) # valor numerico del 4to elemento
146     numero = int(inst[5]) # valor numerico del 6to elemento
147     # Realizamos la resta y actualizamos el registro destino
148     registros[destino] = registros[regi] / numero
149     # Incrementamos el contador de programa para pasar a la siguiente instruccion
150     pc = pc + 1
151
152 # Ver los registros
153 print(registros)
154
155
```

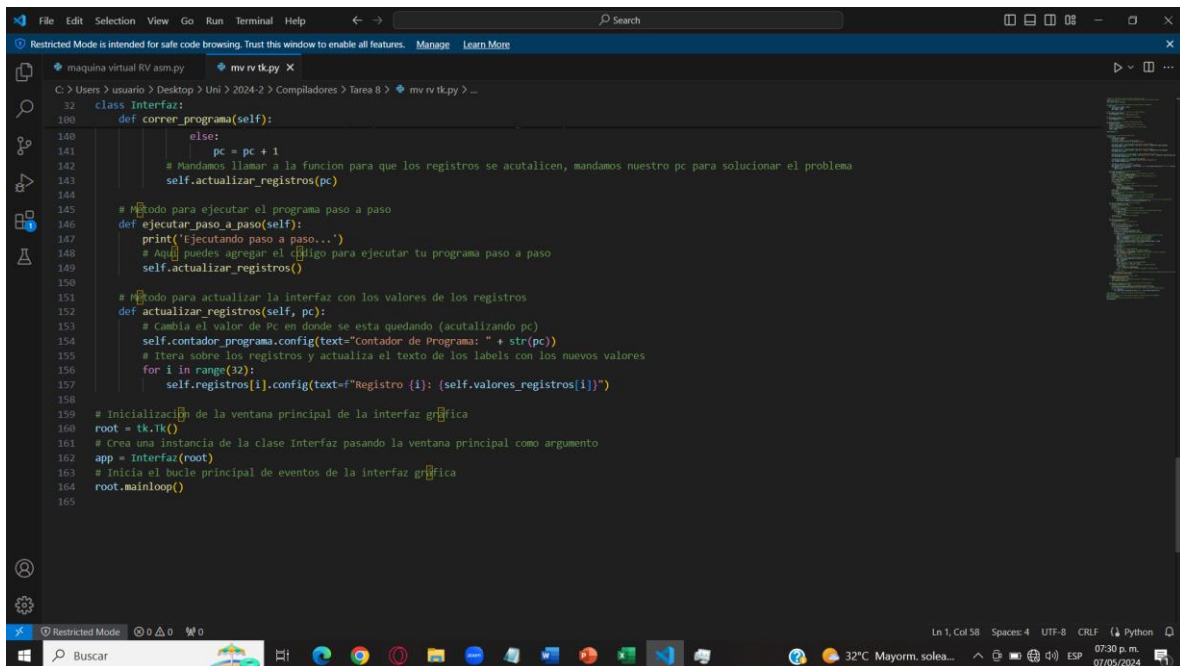
Revise el otro programa, es el mismo, pero con interfaz gráfica, explique cada parte



```
1 # Importa la biblioteca tkinter para crear interfaces gráficas
2 # Importa el módulo filedialog de tkinter para manejar cuadros de dialogo de archivos
3 import tkinter as tk
4 from tkinter import filedialog
5
6 # Creamos la clase etiqueta para almacenar de mejor manera las etiquetas
7 class Etiqueta:
8     def __init__(self, nombre, linea):
9         self.nombre = nombre
10        self.linea = linea
11
12 # Función para verificar si un caracter es un símbolo especial
13 def es_simbolo_especial(caracter):
14     return caracter in "+-*/,%<=>{}[]<>=<=>="
15
16 # Función para verificar si un caracter es un separador
17 def es_separador(caracter):
18     return caracter in " \n\t"
19
20 # Función para tokenizar una cadena y dividirla en tokens
21 def tokenizar(cadena):
22     # Reemplaza las comas por un espacio seguido de una coma
23     cadena = cadena.replace(",", " , ")
24     # Divide la cadena por espacios y devuelve la lista de tokens
25     tokens = cadena.split()
26     return tokens
27
28 instrucciones = []
29
30
31 # Definición de la clase Interfaz para la GUI
32 class Interfaz:
33     def __init__(self, master):
34         self.master = master
35         # Título del programa en la interfaz gráfica
36         self.master.title("Interprete de Ensamblador RISC-V")
37
```

```
File Edit Selection View Go Run Terminal Help
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
maquina virtual RV asm.py mv rv tk.py X
C:\Users\usuario\Desktop> Uni > 2024-2 > Compiladores > Tarea 0 > mv rv tk.py > ...
30
31 # Definición de la clase Interfaz para la GUI
32 class Interfaz:
33     def __init__(self, master):
34         self.master = master
35         # Título del programa en la interfaz grafica
36         self.master.title("Interprete de Ensamblador RISC-V")
37
38         # Botón para cargar el programa usando la función "cargar_programa"
39         self.boton_cargar = tk.Button(master, text="Cargar Programa", command=self.cargar_programa)
40         self.boton_cargar.pack()
41
42         # Botón para correr el programa usando la función "correr_programa"
43         self.boton_correr = tk.Button(master, text="Correr Programa", command=self.correr_programa)
44         self.boton_correr.pack()
45
46         # Botón para ejecutar paso a paso usando la función "ejecutar_paso_a_paso"
47         self.boton_paso_a_paso = tk.Button(master, text="Ejecutar Paso a Paso", command=self.ejecutar_paso_a_paso)
48         self.boton_paso_a_paso.pack()
49
50         # Contador de programa (PC) # REVISAR PUES NO FUNCIONA
51         self.contador_programa = tk.Label(master, text="Contador de Programa: 0")
52         self.contador_programa.pack()
53
54         # Registros (mostrados en programa)
55         self.valores_registros = [0 for _ in range(32)]
56         self.registros = [tk.Label(master, text=f"Registro {i}: {self.valores_registros[i]}") for i in range(32)]
57         for registro in self.registros:
58             registro.pack()
59
60         # Función para cargar un programa desde un archivo
61         def cargar_programa(self):
62             # Abre un cuadro de diálogo para que el usuario seleccione un archivo
63             filename = filedialog.askopenfilename()
64             # Imprime un mensaje indicando que el programa ha sido cargado con éxito
65             print(f'Programa cargado: {filename}')
66
67         # Recorre cada línea del archivo
68         for ren in arch:
69             # Verifica si la longitud es mayor a 2
70             if len(ren) > 2:
71                 # Tokeniza la línea y agrega los tokens a la lista de instrucciones
72                 datos = tokenizar(ren)
73                 instrucciones.append(datos)
74             # Cierra el archivo después de leerlo
75             arch.close()
76
77         # Inicializa una lista vacía para almacenar etiquetas
78         self.etiquetas = []
79         # Itera sobre el número de instrucciones en la lista de instrucciones
80         for c in range(len(instrucciones)):
81             # Verifica si la primera parte de la instrucción es una etiqueta
82             if instrucciones[c][0] == 'X':
83                 # Agrega la etiqueta a la lista de etiquetas junto con su número de línea correspondiente
84                 self.etiquetas.append(etiqueta(instrucciones[c][1], c))
85
86         # Método para obtener la dirección de una etiqueta
87         def get_dir_etiqueta(self, etq):
88             # Inicializa la dirección como -1
89             direccion = -1
90             # Itera sobre cada etiqueta en la lista de etiquetas
91             for e in self.etiquetas:
92                 # Verifica si el nombre de la etiqueta coincide con la etiqueta proporcionada
93                 if e.nombre == etq:
94                     # Retorna el número de línea asociado a la etiqueta
```

```
File Edit Selection View Go Run Terminal Help
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
maquina virtual RV asm.py mv rv tk.py X
C:\Users\usuario\Desktop> Uni > 2024-2 > Compiladores > Tarea 0 > mv rv tk.py > ...
32 class Interfaz:
60 # Función para cargar un programa desde un archivo
61 def cargar_programa(self):
62     # Abre un cuadro de diálogo para que el usuario seleccione un archivo
63     filename = filedialog.askopenfilename()
64     # Imprime un mensaje indicando que el programa ha sido cargado con éxito
65     print(f'Programa cargado: {filename}')
66     # Abre el archivo en modo lectura
67     arch = open(filename, 'r')
68     # Recorre cada línea del archivo
69     for ren in arch:
70         # Verifica si la longitud es mayor a 2
71         if len(ren) > 2:
72             # Tokeniza la línea y agrega los tokens a la lista de instrucciones
73             datos = tokenizar(ren)
74             instrucciones.append(datos)
75         # Cierra el archivo después de leerlo
76         arch.close()
77         # Inicializa una lista vacía para almacenar etiquetas
78         self.etiquetas = []
79         # Itera sobre el número de instrucciones en la lista de instrucciones
80         for c in range(len(instrucciones)):
81             # Verifica si la primera parte de la instrucción es una etiqueta
82             if instrucciones[c][0] == 'X':
83                 # Agrega la etiqueta a la lista de etiquetas junto con su número de línea correspondiente
84                 self.etiquetas.append(etiqueta(instrucciones[c][1], c))
85
86         # Método para obtener la dirección de una etiqueta
87         def get_dir_etiqueta(self, etq):
88             # Inicializa la dirección como -1
89             direccion = -1
90             # Itera sobre cada etiqueta en la lista de etiquetas
91             for e in self.etiquetas:
92                 # Verifica si el nombre de la etiqueta coincide con la etiqueta proporcionada
93                 if e.nombre == etq:
94                     # Retorna el número de línea asociado a la etiqueta
```

```
File Edit Selection View Go Run Terminal Help
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
maquina virtual RV asm.py mv rv tk.py X
C:\Users\usuario\Desktop> Uni > 2024-2 > Compiladores > Tarea 8 > mv rv tk.py > ...
32 class Interfaz:
100 def correr_programa(self):
140     else:
141         pc = pc + 1
142         # Mandamos llamar a la funcion para que los registros se actualicen, mandamos nuestro pc para solucionar el problema
143         self.actualizar_registros(pc)
144
145     # Metodo para ejecutar el programa paso a paso
146     def ejecutar_paso_a_paso(self):
147         print('Ejecutando paso a paso...')
148         # Aqui puedes agregar el codigo para ejecutar tu programa paso a paso
149         self.actualizar_registros()
150
151     # Metodo para actualizar la interfaz con los valores de los registros
152     def actualizar_registros(self, pc):
153         # Cambia el valor de pc en donde se esta quedando (actualizando pc)
154         self.contador_programa.config(text="Contador de Programa: " + str(pc))
155         # Itera sobre los registros y actualiza el texto de los labels con los nuevos valores
156         for i in range(32):
157             self.registros[i].config(text=f'Registro {i}: {self.valores_registros[i]}')
158
159     # Inicialización de la ventana principal de la interfaz grafica
160     root = tk.Tk()
161     # Crea una instancia de la clase Interfaz pasando la ventana principal como argumento
162     app = Interfaz(root)
163     # Inicia el bucle principal de eventos de la interfaz grafica
164     root.mainloop()
165
```

Parte B

Explique en qué consisten las interrupciones (investigue, lea y resuma)

Interrupciones

Una interrupción es cualquier disrupción en la capacidad de la solución de base de datos para prestar servicio a las aplicaciones de usuario. Las interrupciones se pueden clasificar en dos grupos: interrupciones no planificadas e interrupciones planificadas.

Interrupciones imprevistas

- Anomalía de un componente del sistema, incluido el error de hardware o software.
- Acciones administrativas o de aplicación de usuario no válidas, de forma que accidentalmente se descarta una tabla que es necesaria para transacciones críticas de negocio.
- Bajo rendimiento debido a una configuración subóptima o a un hardware o software inadecuado.

Interrupciones planificadas

- Mantenimiento. Algunas actividades de mantenimiento requieren que se realice una interrupción completa; otras actividades de mantenimiento se pueden realizar sin detener la base de datos, pero pueden afectar negativamente al rendimiento. Este último es el tipo más común de interrupción planificada.
- Actualización. La actualización de software o hardware a veces puede requerir una interrupción parcial o total.